embedded OGL protocol, such as a command to create or change the state of a 3D image within an X window) is routed to GLX layer 177. A command interfaced with the DIX 175 is executed by the DIX 175 and potentially by a device dependent layer (DDX) 179, which drives graphical data associated with the executed command through pipeline hardware 166 to frame buffer 65. A command interfaced with GLX layer 177 is transmitted by the GLX layer 177 across LAN 62 to slave pipelines 56-59. One or more of the pipelines 56-59 executes the command and drives graphical data associated with the command to one or more frame buffers 66-69.

In the preferred embodiment, each of slave pipelines 56-59 is configured according to FIG. 6, although other configurations of pipelines 56-59 in other embodiments are possible. As shown by FIG. 6, each slave pipeline 56-59 includes an X server 202, similar to the X server 162 previously described, and an OGL daemon 205. The X server 202 and OGL daemon 205 may be implemented in software, hardware, or a combination thereof, and in the embodiment shown by FIG. 6, the X server 202 and OGL daemon 205 are implemented in software and stored in memory 206. Similar to client 52 and master pipeline 55, each of the slave pipelines 56-59 includes one or more processing elements 181 that communicate to and drive the other elements within the pipeline 56-59 via a local interface 183, which can include one or more buses. Furthermore, an input device 185, for example, a keyboard or a mouse, can be used to input data from a user of the pipeline 56-59, and an output device 187, for example, a display device or a printer, can be used to output data to the user. A disk storage mechanism 192 can be connected to the local interface 183 to transfer data to and from a nonvolatile disk (*e.g.*, magnetic, optical, *etc.*). Each pipeline 56-59 is preferably connected to a LAN interface 196 that allows the pipeline 56-59 to exchange data with the LAN 62.

Similar to X server 162, the X server 202 includes an X server dispatch layer 208, a GLX layer 211, a DIX layer 214, and a DDX layer 216. In the preferred embodiment, each command received by the slave pipelines 56-59 includes 3D graphical data, since the X server 162 of master pipeline 55 executes each X window command that does not include 3D graphical data. The X server dispatch layer 208 interfaces the 2D data of any received commands with DIX layer 214 and interfaces the 3D data of any received commands with GLX layer 211. The DIX and DDX layers 214 and 216 are configured to process or accelerate the 2D data and to drive the 2D data through pipeline hardware 166 to one of the frame buffers 66-69 (FIG. 3).

The GLX layer 211 interfaces the 3D data with the OGL dispatch layer 223 of the OGL daemon 205. The OGL dispatch layer 223 interfaces this data with the OGL DI layer 225. The OGL DI layer 225 and DD layer 227 are configured to process the 3D data and to accelerate or drive the 3D data through pipeline hardware 199 to one of the frame buffers 66-69 (FIG. 3). Thus, the 2D graphical data of a received command is processed or accelerated by the X server 202, and the 3D graphical data of the received command is processed or accelerated by the OGL daemon 205. For a more detailed description of the foregoing process of accelerating 2D data via an X server 202 and of accelerating 3D data via an OGL daemon 205, refer to U.S. patent application serial number 09/138,456.

Preferably, the slave pipelines 56-59, based on inputs from the master pipeline 55, are configured to render 3D images based on the graphical data from master pipeline 55 according to one of three modes of operation: the optimization mode, the super-sampling mode, and the jitter mode. In the optimization mode, each of the slave pipelines 56-59 renders a different portion of a 3D image such that the overall process of rendering the 3D image is faster. In the super-sampling mode, each portion of a 3D image rendered by one or more of the slave pipelines 56-59 is super-sampled in order to increase quality of the 3D image via anti-aliasing. Furthermore, in the jitter mode, each of the slave pipelines 56-59 renders the same 3D image but slightly offsets each rendered 3D image with a different offset value. Then, the compositor 76 averages the pixel data of each pixel for the 3D images rendered by the pipelines 56-59 in order to produce a single 3D image of increased image quality. Each of the foregoing modes will be described in more detail hereafter.

Optimization Mode

Referring to FIG. 3, the operation and interaction of the client 52, the pipelines 55-59, and the compositor 76 will now be described in more detail according to a preferred embodiment of the illustrated environment while the system 50 is operating in the optimization mode. In such an embodiment, the master pipeline 55, in addition to controlling the operation of the slave pipelines 56-59 as described hereinafter, is used to create and manipulate an X window to be displayed by the display device 83. Furthermore, each of the slave pipelines 56-59 is used to render 3D graphical data within a portion of the foregoing X window.

For the purposes of illustrating the aforementioned embodiment, assume that the application 17 issues a function call (*i.e.*, the client 52 via processing element 111 (FIG. 4) executes a function call within the application 17) for creating an X window having a 3D image displayed within the X window. FIG. 7 depicts a more detailed view of the display device 83 displaying such a window 245 on a display device screen 247. In the example shown by FIG. 7, the screen 247 is 2000 pixels by 2000 pixels ("2K x 2K"), and the X window 245 is 1000 pixels by 1000 pixels ("1K x 1K"). The window 245 is offset from each edge of the screen 247 by 500 pixels. Assume that 3D graphical data is to be rendered in a center region 249 of the X window 245. This center region 249 is offset from each edge of the window 245 by 200 pixels in the embodiment shown by FIG. 7.